What Is Claimed Is:

1. A method to facilitate sharing of dynamically compiled native code among multiple tasks of a multitasking virtual machine, comprising:

upon invocation by a task among multiple tasks of a program method not associated with native code, executing a platform-independent code of the program method,

wherein the platform-independent code is shared among multiple tasks, and

wherein an interpreter is re-entrant so that the task can use the interpreter to execute the platform independent code; and

compiling the platform-independent code to one of a task re-entrant native code and a class initialization dependent native code after a first threshold number of executions of the platform-independent code,

wherein a native code is shared among multiple tasks,

wherein the task re-entrant native code includes a task re-entrant class initialization barrier where a class may be used for a first time by the task, and

wherein the class initialization dependent native code includes the task re-entrant class initialization barrier where the class that does not belong to an initialized set associated with the native code may be used for the first time by the task.

2. The method of claim 1, further comprising upon invocation by the task of a program method associated with one version of native code that is task re-entrant, executing the task re-entrant native code.

3. The method of claim 1, further comprising upon invocation by the task of a program method associated with one version of native code that is class initialization dependent, executing the class initialization dependent native code if the task has initialized all classes from the initialized set of the class initialization dependent native code, otherwise, interpreting the platform-independent code of the program method.

4. The method of claim 3, further comprising building task re-entrant native code after a second threshold number of executions by the task of the platform-independent code of the program method associated with class initialization dependent native code.

5. The method of claim 4, further comprising replacing the class initialization dependent native code with the task re-entrant native code.

6. The method of claim 4, further comprising associating the program method with the task re-entrant native code in addition to the class initialization dependent native code.

7. The method of claim 6, further comprising upon invocation by the task of the program method associated with both task re-entrant and class initialization dependent native code, executing the class initialization dependent native code if the task has initialized all classes from the initialized set of the class initialization dependent native code, otherwise, executing the task re-entrant code of the program method.

8. The method of claim 3, further comprising building another class initialization dependent native code after a second threshold number of executions by the task of the platform-independent code of the program method associated with class initialization dependent native code, and replacing initialization dependent native code of a first class with a new class initialization dependent native code, wherein the new class initialization dependent native code is associated with an initialized set different from a first class initialization dependent native code.

9. The method of claim 3, further comprising:

associating a shared representation of the class with a reverse initialized set that records class initialization dependent native code that includes the class in their initialized set; and

associating the task with an array of invocation counters;

wherein an invocation counter of the array of invocation counters with a positive counter value indicates both that the task has not initialized all classes of the initialized set of the class initialization dependent native code of the program method associated with the invocation counter, and a number of invocations of the program method by the task that did not execute the class initialization dependent native code of the program method; and

wherein a specified negative value of the invocation counter indicates that the task has initialized all classes in the initialized set of the class initialization dependent native code of the program method and can execute the class initialization dependent native code upon invocation of the program method.

10. The method of claim 9, further comprising associating the shared representation of the class with the reverse initialized set upon first addition of the

class to the initialized set of the class initialization dependent native code of a program method.

11. The method of claim 9, further comprising adding the class initialization dependent native code that includes the class in its initialized set to the reverse initialized set of the class upon creation of the class initialization dependent native code.

12. The method of claim 9, further comprising upon initialization of the class by the task, setting for each class initialization dependent native code of the reverse initialized set of the class a corresponding invocation counter in the array of invocation counters of the task to the specified negative value if all classes in the initialized set of the class initialization dependent native code, excluding the class, are initialized.

13. The method of claim 9, further comprising:

upon invocation by the task of the program method associated with the class initialization dependent native code, testing a value of the invocation counter of the array of invocation counter of the task that corresponds to the class initialization dependent native code; and

if the value of the invocation counter is negative, executing the class initialization dependent native code.

14. The method of claim 1, wherein the initialized set used to build the class initialization dependent native code for the program method is an intersection of classes already initialized by a first task to reach the first threshold

number of executions of the platform-independent code, and a set of classes whose initialization can be triggered by the program method to be compiled.

15. The method of claim 1, wherein the initialized set used to build the class initialization dependent native code for the program method is a set of all classes whose initialization may be triggered by the program method to be compiled.

16. The method of claim 1, wherein the initialized set used to build class initialization dependent native code for the program method is an intersection of classes already initialized by one or more running tasks and classes whose initialization may be triggered by the program method to be compiled.

17. The method of claim 1, wherein the initialized set used to build a class initialization native code for all program methods of a particular class is identical, whereby the class initialization dependent native code of a first program method of the class can be executed without testing beforehand if an invoking task has initialized all classes of its initialized set when the first program method is invoked from the class initialization dependent native code of a second program method of the class.

18. The method of claim 1, further comprising if a space occupied by all dynamically compiled code exceeds a specified threshold, removing the class initialization dependent native code from the program method when the program method includes task re-entrant code.

19. The method of claim 1, wherein the program method that includes a number of class initialization barriers smaller than a specified threshold are dynamically compiled into task re-entrant native code only.

20. A computer-readable storage medium storing instructions that when executed by a computer cause the computer to perform a method to facilitate sharing of dynamically compiled native code among multiple tasks of a multitasking virtual machine, comprising:

upon invocation by a task among multiple tasks of a program method not associated with native code, executing a platform-independent code of the program method,

wherein the platform-independent code is shared among multiple tasks, and

wherein an interpreter is re-entrant so that the task can use the interpreter to execute the platform independent code; and

compiling the platform-independent code to one of a task re-entrant native code and a class initialization dependent native code after a first threshold number of executions of the platform-independent code,

wherein a native code is shared among multiple tasks,

wherein the task re-entrant native code includes a task re-entrant class initialization barrier where a class may be used for a first time by the task, and

wherein the class initialization dependent native code includes the task re-entrant class initialization barrier where the class that does not belong to an initialized set associated with the native code may be used for the first time by the task.

21. The computer-readable storage medium of claim 20, the method further comprising upon invocation by the task of a program method associated with one version of native code that is task re-entrant, executing the task re-entrant native code.

22. The computer-readable storage medium of claim 20, the method further comprising upon invocation by the task of a program method associated with one version of native code that is class initialization dependent, executing the class initialization dependent native code if the task has initialized all classes from the initialized set of the class initialization dependent native code, otherwise, interpreting the platform-independent code of the program method.

23. The computer-readable storage medium of claim 22, the method further comprising building task re-entrant native code after a second threshold number of executions by the task of the platform-independent code of the program method associated with class initialization dependent native code.

24. The computer-readable storage medium of claim 23, the method further comprising replacing the class initialization dependent native code with the task re-entrant native code.

25. The computer-readable storage medium of claim 23, the method further comprising associating the program method with the task re-entrant native code in addition to the class initialization dependent native code.

26. The computer-readable storage medium of claim 25, the method further comprising upon invocation by the task of the program method associated

with both task re-entrant and class initialization dependent native code, executing the class initialization dependent native code if the task has initialized all classes from the initialized set of the class initialization dependent native code, otherwise, executing the task re-entrant code of the program method.

27. The computer-readable storage medium of claim 22, the method further comprising building another class initialization dependent native code after a second threshold number of executions by the task of the platform-independent code of the program method associated with class initialization dependent native code, and replacing initialization dependent native code of a first class with a new class initialization dependent native code, wherein the new class initialization dependent native code is associated with an initialized set different from a first class initialization dependent native code.

28. The computer-readable storage medium of claim 22, the method further comprising:

associating a shared representation of the class with a reverse initialized set that records class initialization dependent native code that includes the class in their initialized set; and

associating the task with an array of invocation counters;

wherein an invocation counter of the array of invocation counters with a positive counter value indicates both that the task has not initialized all classes of the initialized set of the class initialization dependent native code of the program method associated with the invocation counter, and a number of invocations of the program method by the task that did not execute the class initialization dependent native code of the program method; and

wherein a specified negative value of the invocation counter indicates that the task has initialized all classes in the initialized set of the class initialization dependent native code of the program method and can execute the class initialization dependent native code upon invocation of the program method.

29. The computer-readable storage medium of claim 28, the method further comprising associating the shared representation of the class with the reverse initialized set upon first addition of the class to the initialized set of the class initialization dependent native code of a program method.

30. The computer-readable storage medium of claim 28, the method further comprising adding the class initialization dependent native code that includes the class in its initialized set to the reverse initialized set of the class upon creation of the class initialization dependent native code.

31. The computer-readable storage medium of claim 28, the method further comprising upon initialization of the class by the task, setting for each class initialization dependent native code of the reverse initialized set of the class a corresponding invocation counter in the array of invocation counters of the task to the specified negative value if all classes in the initialized set of the class initialization dependent native code, excluding the class, are initialized.

32. The computer-readable storage medium of claim 28, the method further comprising:

upon invocation by the task of the program method associated with the class initialization dependent native code, testing a value of the invocation counter

of the array of invocation counter of the task that corresponds to the class initialization dependent native code; and

if the value of the invocation counter is negative, executing the class initialization dependent native code.

33. The computer-readable storage medium of claim 20, wherein the initialized set used to build the class initialization dependent native code for the program method is an intersection of classes already initialized by a first task to reach the first threshold number of executions of the platform-independent code, and a set of classes whose initialization can be triggered by the program method to be compiled.

34. The computer-readable storage medium of claim 20, wherein the initialized set used to build the class initialization dependent native code for the program method is a set of all classes whose initialization may be triggered by the program method to be compiled.

35. The computer-readable storage medium of claim 20, wherein the initialized set used to build class initialization dependent native code for the program method is an intersection of classes already initialized by one or more running tasks and classes whose initialization may be triggered by the program method to be compiled.

36. The computer-readable storage medium of claim 20, wherein the initialized set used to build a class initialization native code for all program methods of a particular class is identical, whereby the class initialization dependent native code of a first program method of the class can be executed

without testing beforehand if an invoking task has initialized all classes of its initialized set when the first program method is invoked from the class initialization dependent native code of a second program method of the class.

37. The computer-readable storage medium of claim 20, the method further comprising if a space occupied by all dynamically compiled code exceeds a specified threshold, removing the class initialization dependent native code from the program method when the program method includes task re-entrant code.

38. The computer-readable storage medium of claim 20, wherein the program method that includes a number of class initialization barriers smaller than a specified threshold are dynamically compiled into task re-entrant native code only.

39. An apparatus to facilitate sharing of dynamically compiled native code among multiple tasks of a multitasking virtual machine, comprising:

an execution mechanism that is configured to execute a platform-independent code of a program method upon invocation by a task among multiple tasks of the program method not associated with native code, ,

wherein the platform-independent code is shared among multiple tasks, and

wherein an interpreter is re-entrant so that the task can use the interpreter to execute the platform independent code; and

a compiling mechanism that is configured to compile the platform-independent code to one of a task re-entrant native code and a class initialization dependent native code after a first threshold number of executions of the platform-independent code,

wherein a native code is shared among multiple tasks,

wherein the task re-entrant native code includes a task re-entrant class initialization barrier where a class may be used for a first time by the task, and

wherein the class initialization dependent native code includes the task re-entrant class initialization barrier where the class that does not belong to an initialized set associated with the native code may be used for the first time by the task.

40. The apparatus of claim 39, wherein the execution mechanism is further configured to execute the task re-entrant native code upon invocation by the task of a program method associated with one version of native code that is task re-entrant.

41. The apparatus of claim 39, wherein the execution mechanism is further configured to execute the class initialization dependent native code if the task has initialized all classes from the initialized set of the class initialization dependent native code, otherwise, interpreting the platform-independent code of the program method upon invocation by the task of a program method associated with one version of native code that is class initialization dependent.

42. The apparatus of claim 41, further comprising a code building mechanism that is configured to build task re-entrant native code after a second threshold number of executions by the task of the platform-independent code of the program method associated with class initialization dependent native code.

43. The apparatus of claim 42, further comprising a replacing mechanism that is configured to replace the class initialization dependent native code with the task re-entrant native code.

44. The apparatus of claim 42, further comprising an associating mechanism that is configured to associate the program method with the task re-entrant native code in addition to the class initialization dependent native code.

45. The apparatus of claim 44, wherein the execution mechanism is further configured to execute the class initialization dependent native code if the task has initialized all classes from the initialized set of the class initialization dependent native code upon invocation by the task of the program method associated with both task re-entrant and class initialization dependent native code, otherwise, executing the task re-entrant code of the program method.

46. The apparatus of claim 41, further comprising a code building mechanism that is configured to build another class initialization dependent native code after a second threshold number of executions by the task of the platform-independent code of the program method associated with class initialization dependent native code, and replacing initialization dependent native code of a first class with a new class initialization dependent native code, wherein the new class initialization dependent native code is associated with an initialized set different from a first class initialization dependent native code.

47. The apparatus of claim 41, further comprising:

an associating mechanism that is configured to associate a shared representation of the class with a reverse initialized set that records class initialization dependent native code that includes the class in their initialized set;

wherein the associating mechanism is further configured to associate the task with an array of invocation counters;

wherein an invocation counter of the array of invocation counters with a positive counter value indicates both that the task has not initialized all classes of the initialized set of the class initialization dependent native code of the program method associated with the invocation counter, and a number of invocations of the program method by the task that did not execute the class initialization dependent native code of the program method; and

wherein a specified negative value of the invocation counter indicates that the task has initialized all classes in the initialized set of the class initialization dependent native code of the program method and can execute the class initialization dependent native code upon invocation of the program method.

48. The apparatus of claim 47, wherein the associating mechanism is further configured to associate the shared representation of the class with the reverse initialized set upon first addition of the class to the initialized set of the class initialization dependent native code of a program method.

49. The apparatus of claim 47, further comprising an adding mechanism that is configured to add the class initialization dependent native code that includes the class in its initialized set to the reverse initialized set of the class upon creation of the class initialization dependent native code.

50. The apparatus of claim 47, further comprising a setting mechanism that is configured to set for each class initialization dependent native code of the reverse initialized set of the class a corresponding invocation counter in the array of invocation counters of the task to the specified negative value if all classes in the initialized set of the class initialization dependent native code, excluding the class, are initialized upon initialization of the class by the task.

51. The apparatus of claim 47, further comprising:

a testing mechanism that is configured to test a value of the invocation counter of the array of invocation counter of the task that corresponds to the class initialization dependent native code upon invocation by the task of the program method associated with the class initialization dependent native code; and

wherein the execution mechanism is further configured to execute the class initialization dependent native code if the value of the invocation counter is negative.

52. The apparatus of claim 39, further comprising a code removing mechanism that is configured to remove the class initialization dependent native code from the program method when the program method includes task re-entrant code if a space occupied by all dynamically compiled code exceeds a specified threshold.